02

Porque usar equals o hashCode

Transcripción

[00:00] Hola ¿cómo están? Espero que todos estén bien y vamos a iniciar nuestra clase 14. Nuestra clase 14 está dentro del bloque Métodos Equals y HashCode. Vamos a ver por qué usar Equals o HashCode dentro de nuestras listas. Para esto primero vamos a duplicar nuestra clase 13, vamos a colocar clase 14 y vamos a ver.

[00:29] Aquí estamos trabajando prácticamente con una colección de string. Ahora vamos a crear una clase alumno. Para esto podemos duplicar esta clase aula, colocamos alumno. Aquí tenemos el private String nombre. Colocamos el código que sería el código del alumno. Una vez hecho esto tenemos aquí String código. Tenemos aquí this.codigo = codigo;

[01:12] Vamos a quitar el set, porque quiero que todo sea creado a través de Constructor. Vamos a generar nuestro getter de código y perfecto. Una vez hecho esto, vamos a modificar toda nuestra lista de alumnos, de nuestros alumnos creados. Vamos a colocar un nuevo alumno por ejemplo llamado Claudia Patricia. Vamos a crear nuestros objetos de tipo alumno.

[01:56] Sería alumno, espacio, el código, "001" por ejemplo. Vamos a utilizar esto de aquí en todas nuestras variables. Vamos a colocar aquí "002", sería "003", "004", "005", "006" y "007", James Bond. Una vez hecho esto, vamos a utilizar aquí el alumno, colocar aquí, nuestra colección va a ser ahora de alumnos. Vamos a quitar toda esta parte de aquí y vamos a ver lo siguiente.

[03:03] Recuerdan que estamos viendo, por ejemplo contains, que podíamos ver si un alumno estaba dentro de una lista, pero estábamos comparando String, ahora estamos comparando clases, una clase alumno, podemos hacer lo siguiente, por ejemplo vamos a imprimir System.out.println, vamos a colocar listaAlumnos.contains, y vamos a colocar alumno1.

[03:38] Vamos a ver si este objeto creado está dentro de nuestra lista de objetos. Ejecutamos y suceso. True, está dentro. Pero imaginémonos que llega otro nuevo alumno con el mismo nombre o se quiere registrar con el mismo nombre que el alumno1, entonces entro aquí, el alumnoNuevo = new Alumno (nombre “Luis MIguel”). Vamos a copiar mejor esa parte de aquí, hay que hacer todo igual.

[04:26] Contains alumno y ahora vamos a ejecutar. ¿Qué debería dar? ¿True o false? Vamos a ver, dio false. ¿Por qué? Sucede lo siguiente. Cuando aquí estamos comparando un nuevo objeto, eso quiere decir es otra referencia en memoria Java y este de aquí también es otra referencia en memoria Java. Cuando haces el cointains es otro totalmente diferente. Entonces prácticamente no son iguales.

[04:59] Pero ahora imaginémonos que queramos hacer esto: System.out.println y colocamos (listaAlumnos.equals). Vamos a colocar aquí por ejemplo alumno1 es igual al alumnoNuevo. ¿Qué debería de dar? Dio False también. ¿Por qué? Porque son objetos totalmente diferentes. Para eso nosotros tenemos la opción de sobrescribir el método equals, dar unos arrays.

[05:46] Para eso, nuestra clase alumno vamos, damos unos arrays en nuestro método equals. Aquí es el objeto que estamos recibiendo, en teoría sería nuestro objeto alumno, entonces tenemos que hacer un cast para alumno. Una vez hecho esto, vamos a ver. ¿Por qué queremos comparar? Vamos a compararlos por nombre también. Vamos a colocar aquí alumno.getNombre.

[06:21] Una vez hecho esto, vamos a colocar aquí this.nombre. Entonces nuestro método equals, ¿fue qué cosa? Fue alterado, fue sobrescrito. Ahora ejecutamos y tenemos lo siguiente: True, pero todavía la lista contains continúa siendo false. ¿Pero qué sucede si nosotros en lugar de HashSet colocamos ArrayList?

[07:01] Porque contains también utiliza el equals. Dio True, pero si aquí están igual, está comparando igual, ¿por qué sucede esto? ¿Recuerdan que en la clase anterior también estuvimos viendo poco cómo se crea un HashSet y cómo se crea tal vez una lista, que básicamente sería secuencial y en HashSet no tiene un orden?

[07:28] Sucede lo siguiente. Sucede que el HashSet crea automáticamente un código, que crea un hash, se llamaría. Este hash va colocando en pequeños bloques nuestros objetos para que sean más fácil de ubicar. Porque imaginen que aquí tengamos un millón de objetos hash y tratar de buscar uno va a ser un poco difícil. Entonces ahí es donde él utiliza el hash. ¿Qué sería?

[08:02] El hash es un código del objeto que va a crear y cada vez que tenga un código parecido, lo va a agrupar en el mismo bloque. Por ejemplo aquí tenemos Luis Miguel. Para esto, tenemos ¿qué cosa hacer? Tenemos que sobrescribir nuestra clase hash, nuestro método hashCode. ¿Qué estamos haciendo? Aquí vamos a poner que el this.nombre se convierte en un código, un código hash.

[08:36] Aquí tiene una lógica que ya está prácticamente hecha, no necesitamos hacer nada. Entonces quiere decir que nuestro string va a tener un código hash y cuando nosotros creamos nuestra lista HashSet, él ya va a agruparlo de acuerdo a los códigos que están siendo creados. Nuestros códigos hash, que son específicamente nombre.

[09:06] Entonces cuando ya hagamos la búsqueda, por ejemplo, con el contains va a dar True. ¿Por qué? Porque él ya va a saber en cuál cajita él va entrar para

buscar ese tipo de alumno que yo estoy queriendo, que por ejemplo sería Luis Miguel. Entonces, se hace mucho más fácil. Esa es una diferencia bien grande entre lo que hacemos con un ArrayList y lo que hacemos con un HashSet o con cualquier tipo de hash.

[09:41] Por eso siempre es bueno, si utilizamos equals, tratemos de utilizar también el HashCode. Y si estamos utilizando una comparación con nombre, también que nuestro HashCode sea con un nombre. Porque así se va a hacer mucho más fácil después a nosotros tratar de cambiar, y aquí queremos que sea por ejemplo new ArrayList y va a ejecutar, y perfecto, va a dar true también.

[10:16] Queremos que ahora sea un hash, va a ejecutar y va a dar true también. Pero si le quitamos el hash, prácticamente solo hacemos que esto funcione como cuando sea un ArrayList, porque va a dar false. Entonces queda prácticamente que tenemos que, cuando sobrescribimos equals también sobrescribamos el método HashCode.

[10:58] Y también si estamos utilizando un nombre, también que el HashCode sea nombre, porque si aquí colocamos código y aquí nuestro código puede cambiar, aquí también va a dar false. Entonces va a confundir, es mejor utilizar los dos del mismo tipo que estamos comparando. Eso sería toda nuestra clase 14, nos vemos en la siguiente clase. Muchas gracias.